CCE4510 Sample Code

Getting started with an easy-to-use E² Studio Project

This Application Note describes how to use the CCE4510_sample_app sample code to evaluate the CCE4510 on the CCE4510-EVAL-V3.

Contents

Figures

Tables

1. Required Tools and Hardware

To use the CCE4510_sample_app sample code, the following tools and hardware are required:

1. CCE4510-EVAL-V3 evaluation board
2. Renesas E² Studio version 2023-07 (3.7.0) or higher (RX version)
3. Debug probe (for programming via SCI or FINE interface), for example Renesas E2 or E2lite Emulator
4. 24 V Power Supply
5. USB Cable (Micro USB type B)

2. Erase Pre-Installed IO-Link Master Demonstration Stack

NOTE
If the IO-Link demonstration master stack needs to be reinstalled, the stack needs to be requested from our software partner TMG TE.

The pre-installed IO-Link demonstration master stack comes with a code protection. This prevents any attempt to read, write or debug the board.

To erase the flash and enable the write, read and debug of the CCE4510-EVAL-V3 again, the following steps need to be taken:

Erase Flash using Debug probe (For example: E2 / E2 lite)

1. Connect the debug probe to the CCE4510-EVAL-V3 (X6)
2. Connect the 24 V Power Supply (JP3)
3. Power on the 24 V Power Supply
4. Use the Renesas Flash Programmer (RFP) and set up a new project, using **FINE** in the interface drop-down menu. When clicking **Connect**, a request for an Authentication Code will show.
5. Click **OK** three times consecutively, then click **cancel**. This will erase the flash due to 3 failed attempts of entering the correct Authentication Code.
6. Click **Connect** again to establish the communication. The CCE4510-EVAL-V3 is now ready for programming/debugging.

Erase Flash using USB

1. Connect the USB to the CCE4510-EVAL-V3 (X5).
2. Pull MD low & UB high (JP3) to get into the USB Boot Mode.
3. Connect the 24 V Power Supply (JP3).
4. Power on the 24 V Power Supply. (The CCE4510-EVAL-V3 should now be recognized as **Generic Boot USB Direct** under the **Renesas USB Development Tools** tab of the PCs Device Manager.)
5. Use the Renesas Flash Programmer (RFP) and set up a new project using **USB direct** as tool. When clicking **Connect**, a request for an Authentication Code will show.
6. Click **OK** three times consecutively, then click **cancel**. This will erase the flash due to 3 failed attempts of entering the correct Authentication Code.
7. Click the **RESET** switch of the CCE4510-EVAL-V3.
8. Click **Connect** again to establish the communication. The CCE4510-EVAL-V3 is now ready for programming / debugging.
9. Optional: if you directly want to flash the board again, click the **RESET** button once again and flash the board with the desired file.

3. Importing the CCE4510_sample_app into E² Studio

To import the CCE4510_sample_app sample code, it is recommended to use the **import wizard** of E² Studio. The **import wizard** can be found in the **File** menu:

File > Import… > General > Existing Projects into Workspace

After starting the **import wizard**:

1. Click **Next** and search for the **CCE4510_sample_app** folder on your hard drive.
2. Select the **CCE4510_sample_app** and click **Finish**.

The CCE4510_sample_app project should now be visible in the **Project Explorer**.

To finish the import process, double-click on **CCE4510_sample_app.scfg**.

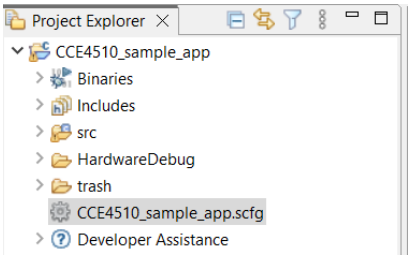

Figure 1. Project explorer view

This opens the smart configurator view, which can be used to check that all necessary software components are available and updated.

To do so, click on the **Components** tab. Figure 2 shows how the **Components** tab looks when all components are correctly installed.

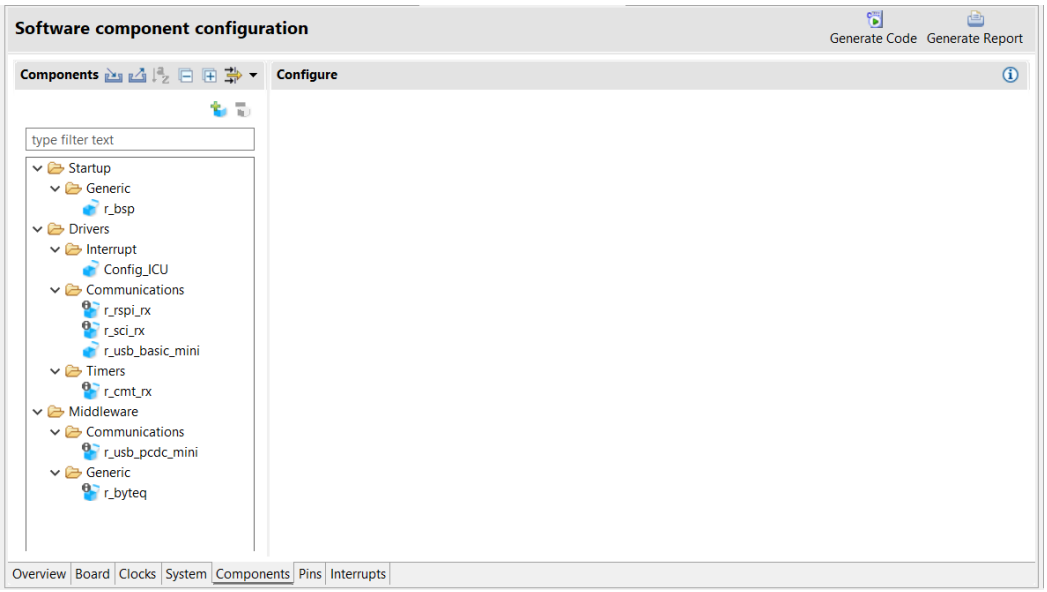

Figure 2. Components tab

Any component can be updated via the context menu of the respective component. After updating all components, click on **Generate Code**.

The CCE4510_sample_app code is now ready to be build.

4. Project Description

The CCE4510_sample_app is designed to easily evaluate the CCE4510 IO-Link Master Transceiver without the need to set up a new project. The user only needs to adapt the CCE4510_sample_app.c file, located in the **src** folder of the project, to control the CCE4510.

It is possible to control the CCE4510 either via SPI or via UART interface.

4.1 Functions Overview

Table 1. Functions overview

Function	Description
cce4510_spi_init	Initializes the SPI interface to communicate with the CCE4510
cce4510_register_read	Reads the selected register of the CCE4510
cce4510_register_write	Writes to the selected register of the CCE4510
cce4510_configure_sio	Configures the CCE4510 to be used in SIO mode
cce4510_dump_register	Reads the selected register and prints its value to the console
cce4510_dump_registers	Reads all registers and prints their values to the console
cce4510_pin_set	Sets the selected pin high or low
cce4510_uart_init	Initializes the UART interface to communicate with the CCE4510
cce4510_uart_send	Sends data over the UART interface
usb_init	Initializes the USB interface to communicate with the host PC
printf	Prints outputs to the console
Delay	Delays the program

4.2 Description of Functions

4.2.1 cce4510_spi_init

Initializes the SPI interface to communicate with the CCE4510. This needs to be called once before starting communication.

Format

```
Rspi err t cce4510 spi init()
```

Parameters

None

4.2.2 cce4510_register_read

Reads the selected register of the CCE4510. All register definitions can be found in cce4510.h.

Format

```
rspi err t cce4510 register read(E CCE4510 REG ADDR t address,
                                 uint8 t *reg value,
                                 uint8_t *p_status)
```

Parameters

address	Address of the register
reg_value	Value of the register
p_status	MISO status nibble

4.2.3 cce4510_register_write

Writes to the selected register of the CCE4510. All register definitions can be found in cce4510.h.

Format

```
rspi err t cce4510 register write(E CCE4510 REG ADDR t address,
                                  uint8_t *reg_value,
                                  uint8 t *p status)
```

Parameters

address	Address of the register
reg_value	Value of the register
p_status	MISO status nibble

4.2.4 cce4510_configure_sio

Configures the CCE4510 to be used in SIO mode

Format

```
Rspi_err_t cce4510_configure_sio();
```

Parameters

None

4.2.5 cce4510_dump_register

Reads the selected register and prints its value to the console. The **name** parameter defines the printed name in the console.

Format

```
cce4510 dump register(char *name,
                      E_CCE4510_REG_ADDR_t address);
```

Parameters

name	Name of the register
address	Address of the register

4.2.6 cce4510_dump_registers

Reads all registers and prints their values to the console

Format

```
cce4510 dump registers();
```

Parameters

None

4.2.7 cce4510_pin_set

Sets the selected pin high or low. Selectable pins are CQ1, CQ2, LP1 and LP2

Format

```
cce4510 pin set(E CCE4510 CHANNEL t channel,
                E_CCE4510_PIN_t pin,
                E CCE4510 PIN LEVEL t level);
```

Parameters

channel	Selects the channel (Channel 1 or Channel 2)
pin	Selects the pin (CQ1, CQ2, LP1 or LP2)
level	Expected level (high or low)

4.2.8 cce4510_uart_init

Initializes the UART interface to communicate with the CCE4510. This needs to be called once before starting communication.

Format

```
sci_err_t cce4510_uart_init(void);
```

Parameters

None

4.2.9 cce4510_uart_send

Sends data over the UART interface

Format

```
Sci_err_t cce4510_uart_send(E_CCE4510_CHANNEL_t channel,
                            uint8 t *p src,
                            uint16_t const length);
```

Parameters

channel	Selects the channel (Channel 1 or Channel 2)
*p_src	Buffer to send
length	Length of the buffer

4.2.10 usb_init

Initializes the USB interface to communicate with the host PC. This needs to be called once before starting communication with the host PC.

Format

```
usb_init(void)
```

Parameters

None

4.2.11 printf

Prints outputs to the console. Standard stdio printf command.

Format

```
printf (const char *__restrict, ...)
```

Parameters

*__restrict see standard stdio printf command description

4.2.12 delay

Static delay before executing the next command.

Format

```
delay(uint32_t us)
```

Parameters

us	delay time in microseconds

4.3 CCE4510_sample_app.c Overview

The CCE4510_sample_app.c file contains two use case examples (internal macros), one for SPI controlled communication and one for UART controlled communication. The user can use and modify these examples as needed or completely develop their own example code in the **main** function.

To choose between the two examples, the user needs to modify the CCE4510_SAMPLE_APP_PHY_COMM_MODE internal macro (see Figure 3)

```
33  #define CCE4510_SAMPLE_APP_PHY_COMM_MODE_UART     (1U)
34  #define CCE4510_SAMPLE_APP_PHY_COMM_MODE_SPI      (2U)
35
36  #define CCE4510_SAMPLE_APP_PHY_COMM_MODE          (CCE4510_SAMPLE_APP_PHY_COMM_MODE_UART)
```

Figure 3. Internal comm mode macro

Changing line 36 from CCE4510_SAMPLE_APP_PHY_COMM_MODE_UART to CCE4510_SAMPLE_APP_PHY_COMM_MODE_SPI will switch the used example from UART to SPI.

Rebuild the project after changes have been made.

The examples are called in the main function, depending on what mode is chosen (see Figure 4).

```
296  void main(void)
297  {
298
299  #if CCE4510_SAMPLE_APP_PHY_COMM_MODE == CCE4510_SAMPLE_APP_PHY_COMM_MODE_SPI
300
301      spi_main();
302
303  #elif CCE4510_SAMPLE_APP_PHY_COMM_MODE == CCE4510_SAMPLE_APP_PHY_COMM_MODE_UART
304
305      uart_main();
306
307  #else
308
309  #error "Unknown PHY communication mode selected!"
310
311  #endif
312  }
```

Figure 4. Main function

4.3.1 SPI Slave Select Configuration

To switch between the SPI configuration of the two CCE4510 transceivers (IC1 and IC2), the SPI initialization configuration needs to be adapted.

The SPI initialization can be found in the cce4510.c file.

```
201  rspi_err_t cce4510_spi_init()
202  {
203      rspi_err_t ret;
204      rspi_chnl_settings_t s_rspi_settings;
205
206      s_rspi_settings.bps_target = CCE4510_SPI_BPS_TARGET;
207      s_rspi_settings.gpio_ssl = RSPI_IF_MODE_4WIRE;
208      s_rspi_settings.master_slave_mode = RSPI_MS_MODE_MASTER;
209      s_rspi_settings.tran_mode = RSPI_TRANS_MODE_SW;
210
211      g_spi_handle.command.bit_length = RSPI_SPCMD_BIT_LENGTH_8;
212      g_spi_handle.command.bit_order = RSPI_SPCMD_ORDER_MSB_FIRST;
213      g_spi_handle.command.br_div = RSPI_SPCMD_BR_DIV_2;
214      g_spi_handle.command.clock_delay = RSPI_SPCMD_CLK_DLY_1;
215      g_spi_handle.command.cpha = RSPI_SPCMD_CPHA_SAMPLE_ODD;
216      g_spi_handle.command.cpol = RSPI_SPCMD_CPOL_IDLE_LO;
217      g_spi_handle.command.next_delay = RSPI_SPCMD_NEXT_DLY_1;
218      g_spi_handle.command.ssl_assert = RSPI_SPCMD_ASSERT_SSL0;
219      g_spi_handle.command.ssl_neg_delay = RSPI_SPCMD_SSL_NEG_DLY_1;
220      g_spi_handle.command.ssl_negate = RSPI_SPCMD_SSL_KEEP;
221
222      IPR(RSPI0, SPEI0) = 15U;
223      IPR(RSPI0, SPRI0) = 15U;
224      IPR(RSPI0, SPTI0) = 15U;
225      IPR(RSPI0, SPII0) = 15U;
226
227
228      R_RSPI_PinSet_RSPI0();
229      ret = R_RSPI_Open(0U, &s_rspi_settings, g_spi_handle.command, cb_rspi, &(g_spi_handle.rspi));
230
231
232      return ret;
233  }
```

Figure 5. SPI initialization

Line 218 determines which SPI slave select line is used when communicating.

Table 2 shows the parameter values to switch between the different CCE4510 ICs.

Table 2. SPI SSL assertion

Parameter for SSL assertion	Selected CCE4510 IC
RSPI_SPCMD_ASSERT_SSL0	IC1
RSPI_SPCMD_ASSERT_SSL2	IC2

4.3.2 UART Example Code

The UART example code sends example data via the UART interface, while using the SPI interface for configuration of the CCE4510. After USB, SPI and UART initialization, the CCE4510 is configured to SIO mode and the current register values are printed to the console.

The example data is then sent once every second.

NOTE
Additional cable connections are required when using the UART example code! See section 5.

Figure 6 shows the UART example code.

```
void uart_main(void)
{
    uint8_t example_data[] = {0xCA, 0xFE, 0xCA, 0xFE, 0xCA, 0xFE};

    usb_init();

    cce4510_spi_init();    /* SPI interface is needed for configuration. */

    cce4510_uart_init();

    printf("Configuring PHY for SIO mode...\r\n");

    cce4510_configure_sio();

    printf("Done\r\n");

    cce4510_dump_registers();

    while(1U)
    {
        printf("Sending example data...\r\n");
        cce4510_uart_send(E_CCE4510_CHANNEL_1, example_data, sizeof(example_data));
        cce4510_uart_send(E_CCE4510_CHANNEL_2, example_data, sizeof(example_data));
        delay(DELAY_1000_MS);
    }

}
#endif
```

Figure 6. UART example code

4.3.3 SPI Example Code

The SPI example code uses the SPI interface to toggle CQ and LP, and to trigger the automatic wake-up sequence of the CCE4510.

After USB and SPI initialization, the CCE4510 is configured to SIO mode and the current register values will be printed to the console. CQ1 and CQ2 are toggled for ten times, LP1 and LP2 are toggled for ten times, followed by triggering the automatic wake-up sequence once every second (see Figure 7, Figure 8 and Figure 9)

```
115  void spi_main(void)
116  {
117      uint8_t status;
118      uint8_t rev;
119      uint8_t register_value;
120      uint8_t i;
121
122      usb_init();
123
124      cce4510_spi_init();
125
126      printf("Retrieving revision code from the PHY...\r\n");
127
128      cce4510_register_read(E_CCE4510_REG_ADDR_REV, &register_value, &status);
129
130      printf("Revision: 0x%02X\r\n", register_value);
131
132      delay(DELAY_1000_MS);
133
134      printf("Configuring PHY for SIO mode...\r\n");
135
136      cce4510_configure_sio();
137
138      printf("Done.\r\n");
139
140      delay(DELAY_1000_MS);
141
142      printf("Reading PHY registers current values...\r\n");
143
144      cce4510_dump_registers();
145
146      delay(DELAY_1000_MS);
```

Figure 7. SPI example code initialization

```
197         printf("Toggling CQ pins...\r\n");
198
199         for(i = 0U; i < 10U; i++)
200         {
201             printf("CQ --> LOW\r\n");
202             cce4510_pin_set(E_CCE4510_CHANNEL_1, E_CCE4510_PIN_CQ, E_CCE4510_PIN_LEVEL_LOW);
203             cce4510_pin_set(E_CCE4510_CHANNEL_2, E_CCE4510_PIN_CQ, E_CCE4510_PIN_LEVEL_LOW);
204
205             delay(DELAY_1000_MS);
206
207
208             printf("CQ --> HIGH\r\n");
209             cce4510_pin_set(E_CCE4510_CHANNEL_1, E_CCE4510_PIN_CQ, E_CCE4510_PIN_LEVEL_HIGH);
210             cce4510_pin_set(E_CCE4510_CHANNEL_2, E_CCE4510_PIN_CQ, E_CCE4510_PIN_LEVEL_HIGH);
211
212             delay(DELAY_1000_MS);
213         }
214
215         printf("Toggling L+ pins...\r\n");
216
217         for(i = 0U; i < 10U; i++)
218         {
219             printf("L+ --> LOW\r\n");
220             cce4510_pin_set(E_CCE4510_CHANNEL_1, E_CCE4510_PIN_LP, E_CCE4510_PIN_LEVEL_LOW);
221             cce4510_pin_set(E_CCE4510_CHANNEL_2, E_CCE4510_PIN_LP, E_CCE4510_PIN_LEVEL_LOW);
222
223             delay(DELAY_1000_MS);
224
225
226             printf("L+ --> HIGH\r\n");
227             cce4510_pin_set(E_CCE4510_CHANNEL_1, E_CCE4510_PIN_LP, E_CCE4510_PIN_LEVEL_HIGH);
228             cce4510_pin_set(E_CCE4510_CHANNEL_2, E_CCE4510_PIN_LP, E_CCE4510_PIN_LEVEL_HIGH);
229
230             delay(DELAY_1000_MS);
231         }
```

Figure 8. SPI CQ and LP toggle

```
233         while(1)
234         {
235             printf("Triggering automatic wake-up sequence...\r\n");
236
237             cce4510_register_read(E_CCE4510_REG_ADDR_SIO2, &register_value, &status);
238             register_value = register_value | CCE4510_REG_SIOX_WURQ_ON_STR_STP;
239             cce4510_register_write(E_CCE4510_REG_ADDR_SIO2, &register_value, &status);
240
241             cce4510_register_read(E_CCE4510_REG_ADDR_STAT, &register_value, &status);
242
243             cce4510_register_read(E_CCE4510_REG_ADDR_FHC2, &register_value, &status);
244             register_value |= CCE4510_REG_FHCX_RST;
245             cce4510_register_write(E_CCE4510_REG_ADDR_FHC2, &register_value, &status);
246
247             cce4510_configure_sio();
248
249             delay(DELAY_1000_MS);
250         }
```

Figure 9. SPI automatic wake-up

4.4 Debugging

To start debugging the CCE4510 sample code, perform the following steps:

1. Connect the debug probe (For example: Renesas E2 lite) to the CCE4510-EVAL-V3.
2. Connect the USB cable from the host PC to the CCE4510-EVAL-V3.
3. Optional: Connect UART interface (see section 5).
4. Connect the 24 V power supply to the CCE4510-EVAL-V3 and power on.
5. Click on **Run** > **Debug Configurations**… and double click on **Renesas GDB Hardware Debugging** to create a debug configuration.
6. In the **Debugger** tab of the created debug configuration, choose the used debugger (**Debug hardware**),for example: **E2 Lite (RX)**.
7. In the **Debugger** tab, switch to the **Connections Settings** tab and change the **Extal Frequency[MHz]** to **12** and choose **No** at **Power Target From the Emulator (MAX 200mA)**.
8. Click **Apply**.
9. Start debugging. The sample code will stop at two break points. Click on **Resume (F8)** to continue until the host PC recognizes the USB Serial Device as **COM** port (for example: **COM9**). The sample code pauses until a terminal is opened.
10. Open a terminal console in E² Studio (**Ctrl+Alt+Shift+T**) and choose **Serial Terminal** with a baud rate of 9600.

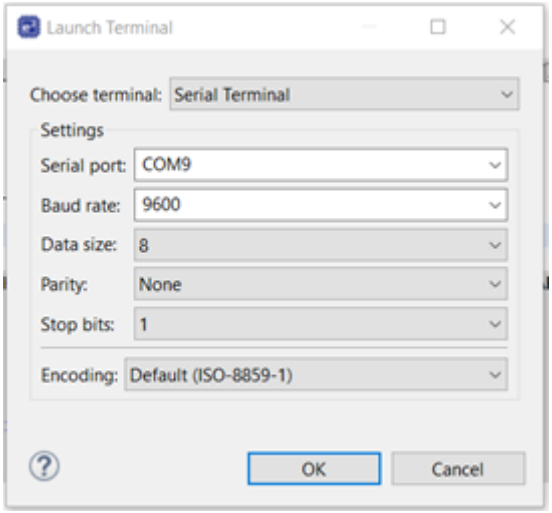

Figure 10. Project explorer view

It is also possible to use any other suitable terminal emulator.

11. The sample code resumes automatically as soon as the terminal is started, and the chosen example is executed.

5. Hardware Connections for UART Mode

When using the UART interface to control the output of the CCE4510 IO-Link interface, additional cable connections must be made.

Figure 11 shows the available connectors for CCE4510-EVAL-V3.

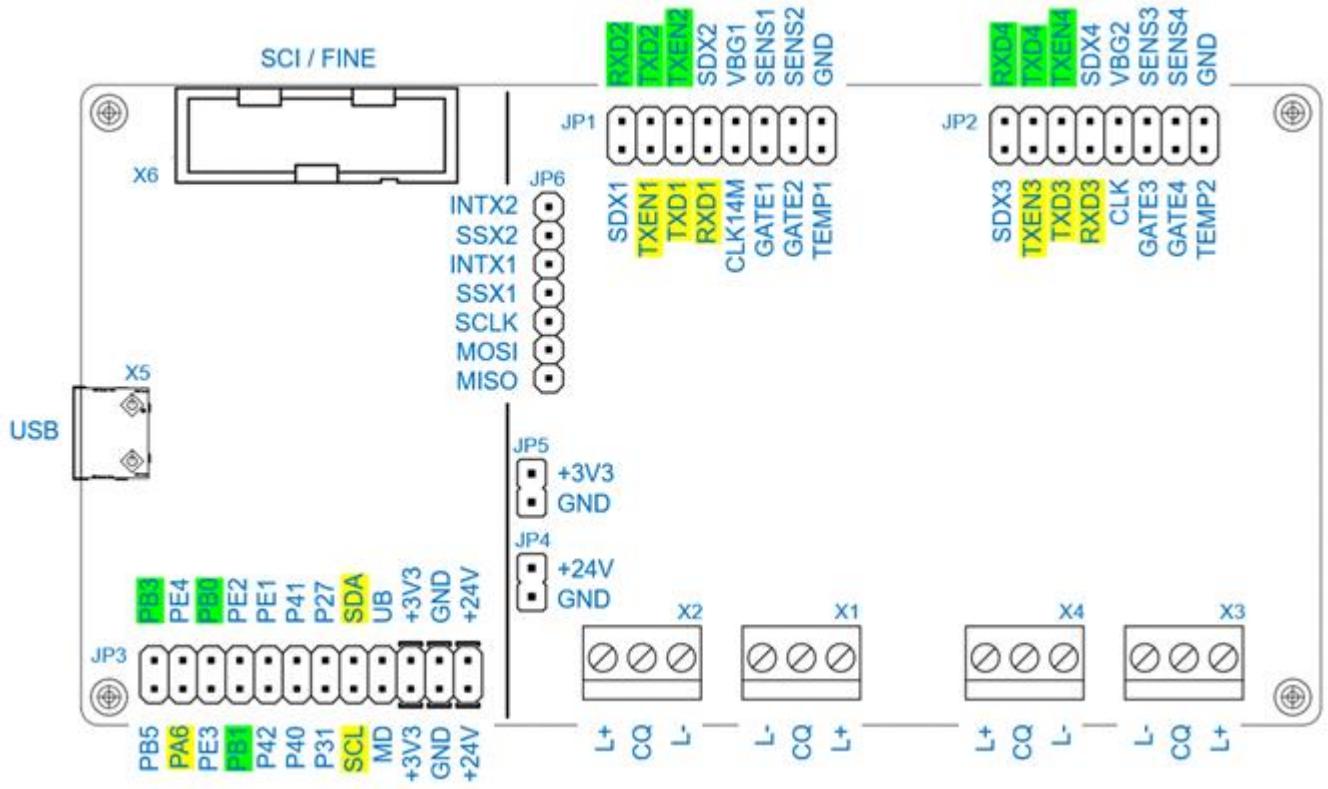

Figure 11. Connectors CCE4510-EVAL-V3

Two UART interfaces are implemented in the sample code. Table 3 shows the required connections to use the UART interfaces.

Table 3. UART connections

Description	JP3	JP1 or JP2
UART1 TXD	SDA (Pin 16)	TXD1 or TXD3
UART1 RXD	SCL (Pin 15)	RXD1 or RXD3
UART1 TXEN	PA6 (Pin 3)	TXEN1 or TXEN3
UART2 TXD	PB1 (Pin 7)	TXD2 or TXD4
UART2 RXD	PB0 (Pin 6)	RXD2 or RXD4
UART2 TXEN	PB3 (Pin 2)	TXEN2 or TXEN4

6. Revision History

Revision	Date	Description
01.10	October 25, 2024	Updated Header
01.00	July 02, 2024	First version.

STATUS DEFINITIONS

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

ROHS COMPLIANCE

Renesas Electronics' suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.